Tugas 5

PEMROGRAMAN MOBILE TEORI

RANGKUMAN

FLUTTER 56 - 60

Kelas I1

187221048 – Muhammad Hanif A S
187221052 – Muhammad Naufal Rizky A W

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS AIRLANGGA

2024

RANGKUMAN DAN IMPLEMENTASI

FLUTTER 56

Authentication

Pada Flutter 56 ini berisikan konsep dasar tentang Authentication yang mengacu pada cara pengidentifikasian dan otorisasi pengguna dalam aplikasi Flutter. Ini melibatkan proses otentikasi, yang umumnya mencakup registrasi pengguna baru, login, dan pengelolaan sesi pengguna. Flutter menyediakan berbagai cara untuk mengimplementasikan sistem otentikasi, termasuk penggunaan penyedia otentikasi pihak ketiga seperti Firebase atau OAuth, serta autentikasi khusus server. Proses otentikasi umumnya melibatkan penggunaan token atau kunci yang unik untuk mengidentifikasi dan memverifikasi pengguna, dan sering kali melibatkan proses enkripsi untuk melindungi informasi sensitif. Dengan menggunakan konsep dasar ini, pengembang Flutter dapat membuat aplikasi yang aman dan mematuhi kebutuhan keamanan pengguna.

FLUTTER 57

Login & Register

Flutter 57 membahas konsep dasar tentang otentikasi dalam aplikasi Flutter, dengan fokus pada proses login dan registrasi menggunakan Firebase Authentication. Ini melibatkan penggunaan Firebase sebagai backend untuk mengelola otentikasi pengguna. Proses login melibatkan pengguna memasukkan kredensial untuk mengakses aplikasi, sementara proses registrasi memungkinkan pengguna untuk membuat akun baru dengan memberikan informasi dasar. Pentingnya validasi input dalam proses ini disoroti, bersama dengan pengelolaan status aplikasi terkait dengan otentikasi.

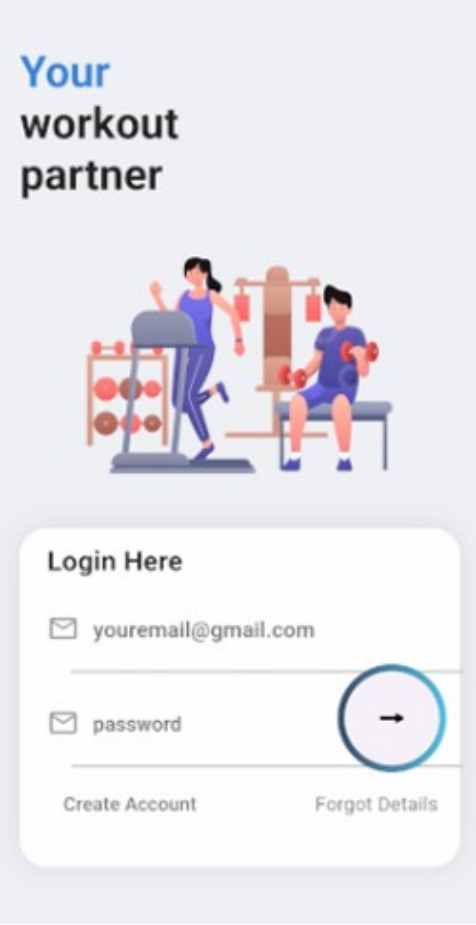
Implementasi

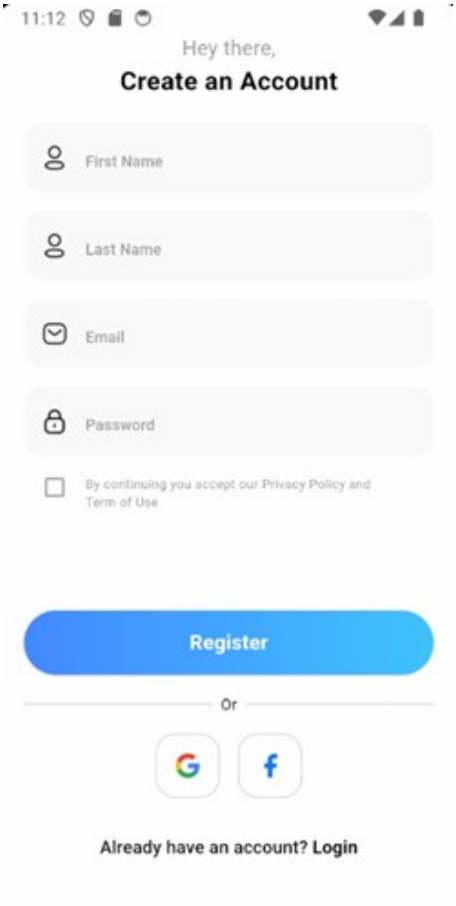
```dart

import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import 'package:flutter/widgets.dart';
import 'package:flutter_application_1/common_widget/form_container_widget.dart';
import 'package:flutter_application_1/common_widget/round_button.dart';
import 'package:flutter_application_1/common_widget/round_textfield.dart';
import 'package:flutter_application_1/login/home.dart';
import 'package:flutter_application_1/user_auth/firebase_auth_services.dart';
import 'package:firebase_auth/firebase_auth.dart';

import '../common_widget/theformfield.dart';
import 'complete_profile_view.dart';

class SignUpView extends StatefulWidget {
  const SignUpView({super.key});

  @override
  State<SignUpView> createState() => _SignUpViewState();
}

class _SignUpViewState extends State<SignUpView> {

  String email = "", password = "", name = "";
  TextEditingController namecontroller = new TextEditingController();
  TextEditingController passwordcontroller = new TextEditingController();
  TextEditingController mailcontroller = new TextEditingController();

  final _formkey = GlobalKey<FormState>();

  registration() async {
    if (password != null&& namecontroller.text!=""&& mailcontroller.text!="") {
```

```dart
  bool isCheck = false;
  @override
  Widget build(BuildContext context) {
    var media = MediaQuery.of(context).size;
    return Scaffold(
      backgroundColor: Colors.white,
      body: SingleChildScrollView(
        child: SafeArea(
          child: Padding(
            padding: const EdgeInsets.symmetric(horizontal: 20),
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.center,
              children: [
                SizedBox(height: media.width * 0.05,),
                Text(
                  "Hey there,",
                  style: TextStyle(color: Colors.grey, fontSize: 16),
                ), // Text

                Text(
                  "Create an Account",
                  style: TextStyle(
                      color: Colors.black,
                      fontSize: 20,
                      fontWeight: FontWeight.w700), // TextStyle
                ), // Text

                SizedBox(height: media.width * 0.1,),

```

```
                    Text(
                      "Already have an account? ",
                      style: TextStyle(
                        color: Colors.black,
                        fontSize: 14,
                      ), // TextStyle
                    ), // Text
                    Text(
                      "Login",
                      style: TextStyle(
                          color: Colors.black,
                          fontSize: 14,
                          fontWeight: FontWeight.w700), // TextStyle
                    ) // Text
                  ],
                ), // Row
              ), // TextButton
              SizedBox(
                height: media.width * 0.04,
              ), // SizedBox
            ],
          ), // Column
        ), // Padding
      ), // SafeArea
    ), // SingleChildScrollView
  ); // Scaffold
 }

}
```

Output





FLUTTER 58

Error Handling

Pada Flutter 58, fokusnya adalah pada Authentication (autentikasi) dan Error Handling (penanganan kesalahan). Autentikasi memverifikasi identitas pengguna, dengan opsi seperti Firebase Authentication. Error Handling penting untuk mengelola situasi kesalahan seperti koneksi internet terputus. Flutter menyediakan alat untuk menangani ini, termasuk try-catch untuk kesalahan sinkron dan widget seperti FutureBuilder untuk kesalahan asynchron.

Implementasi

```
33        try {
34          UserCredential userCredential = await FirebaseAuth.instance
35              .createUserWithEmailAndPassword(email: email, password: password);
36          ScaffoldMessenger.of(context).showSnackBar(SnackBar(
37              content: Text(
38            "Registered Successfully",
39            style: TextStyle(fontSize: 20.0),
40          ))); // Text // SnackBar
41          // ignore: use_build_context_synchronously
42          Navigator.push(
43              context, MaterialPageRoute(builder: (context) => Home()));
44        } on FirebaseAuthException catch (e) {
45          if (e.code == 'weak-password') {
46            ScaffoldMessenger.of(context).showSnackBar(SnackBar(
47                backgroundColor: Colors.orangeAccent,
48                content: Text(
49                  "Password Provided is too Weak",
50                  style: TextStyle(fontSize: 18.0),
51                ))); // Text // SnackBar
52          } else if (e.code == "email-already-in-use") {
53            ScaffoldMessenger.of(context).showSnackBar(SnackBar(
54                backgroundColor: Colors.orangeAccent,
55                content: Text(
56                  "Account Already exists",
57                  style: TextStyle(fontSize: 18.0),
58                ))); // Text // SnackBar
59          }
60        }
61      }
62    }
```

FLUTTER 59 Authentication - Token Handling

Firebase menggunakan token untuk memverifikasi identitas pengguna saat mereka melakukan request. Token ini diperoleh saat login atau signup dan berfungsi sebagai pengenal unik untuk setiap pengguna. Masa berlaku token terbatas, dan perlu diperbarui sebelum kadaluwarsa agar akses ke Firebase tidak terputus. Aplikasi perlu mengecek keabsahan token dengan mencocokkan tanggal kedaluwarsa dan nilai awal token sebelum mengirim request ke Firebase.

Penjelasan:

- Token sebagai mekanisme autentikasi: Token berperan sebagai kunci digital yang membuktikan identitas pengguna kepada Firebase.
- Penerimaan token: Token diberikan kepada pengguna setelah login atau signup yang sukses.

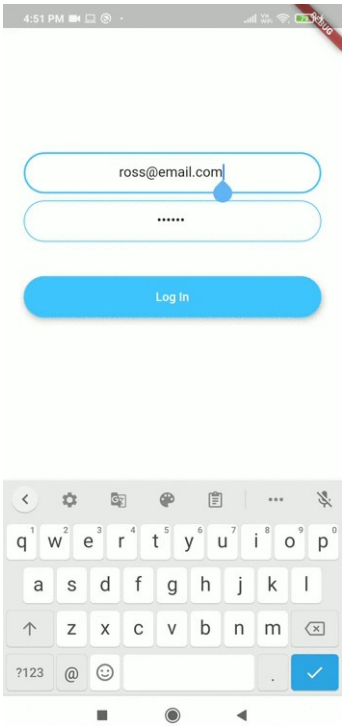
- Fungsi token: Token digunakan untuk mengidentifikasi dan mengotentikasi pengguna pada setiap request ke Firebase.
- Masa berlaku token: Token memiliki batas waktu, setelah itu menjadi tidak valid.
- Memeriksa keabsahan token: Penting untuk memastikan token belum kadaluwarsa sebelum digunakan untuk request ke Firebase.
- Cara memeriksa keabsahan token: Token diverifikasi dengan mencocokkan tanggal kedaluwarsa dan nilai awal token.

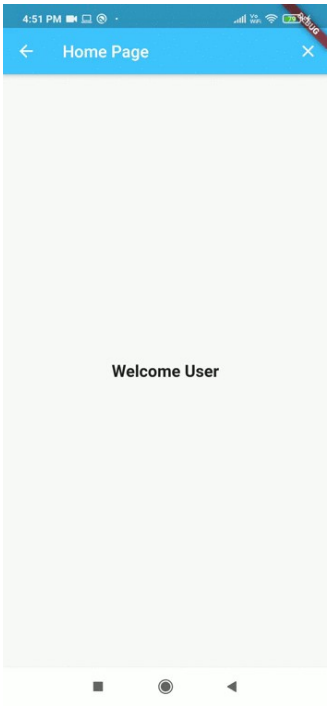
CARA KERJA :

1. Penerimaan Token: Setelah pengguna berhasil melakukan proses autentikasi (misalnya, login), server akan menghasilkan token autentikasi dan mengirimkannya kembali ke aplikasi Flutter.

2. Penyimpanan Token: Token autentikasi disimpan secara aman dalam aplikasi Flutter. Ini dapat dilakukan dengan menggunakan penyimpanan lokal seperti Shared Preferences atau penyimpanan aman lainnya yang disediakan oleh Flutter.

3. Penggunaan Token: Setiap kali aplikasi membuat permintaan ke server yang memerlukan autentikasi, token autentikasi disertakan dalam header permintaan. Server kemudian memvalidasi token tersebut untuk memastikan bahwa pengguna memiliki akses yang diotorisasi.





FLUTTER 60 Authentication - User ID Handling

Proses autentikasi di aplikasi Flutter melibatkan **penanganan ID pengguna**, sebuah mekanisme untuk mengelola dan mengidentifikasi setiap pengguna yang telah berhasil login. Setiap pengguna mendapatkan **ID unik** yang berfungsi sebagai penanda identitas mereka di dalam sistem.

Penjelasan:

- **Penanganan ID pengguna:** Proses ini mencakup berbagai langkah untuk mengelola dan memanfaatkan ID pengguna dalam konteks autentikasi.
- **Identifikasi pengguna:** ID unik diberikan kepada setiap pengguna untuk membedakan mereka satu sama lain dalam sistem.
- **Manfaat ID pengguna:** ID pengguna bermanfaat untuk berbagai keperluan, seperti:
    - **Pelacakan aktivitas pengguna**
    - **Personalisasi pengalaman pengguna**
    - **Penyimpanan data pengguna**
    - **Pengamanan akses ke sumber daya aplikasi**

CARA KERJA :

1. Penciptaan ID Pengguna: Setelah pengguna berhasil melakukan proses autentikasi (misalnya, login), mereka diberikan sebuah ID pengguna yang unik.

2. Penyimpanan ID Pengguna: ID pengguna disimpan dalam penyimpanan lokal seperti Shared Preferences, database lokal (misalnya, SQLite), atau disimpan bersama informasi pengguna di server.

3. Penggunaan ID Pengguna: ID pengguna digunakan untuk mengidentifikasi pengguna saat membuat permintaan ke server atau saat menyimpan data terkait pengguna dalam aplikasi.

Kesimpulan:

Penanganan ID pengguna merupakan aspek penting dalam autentikasi aplikasi Flutter. Dengan mengelola ID pengguna secara efektif, aplikasi dapat memberikan pengalaman yang lebih aman dan personal bagi para penggunanya.

```
import 'package:flutter/material.dart';
import 'package:flutter_login/flutter_login.dart';
import 'package:provider/provider.dart';

import '../providers/auth.dart';

const users = const {
  'dribbble@gmail.com': '12345',
  'hunter@gmail.com': 'hunter',
};
```

```dart
class LoginPage extends StatefulWidget {
  @override
  _LoginPageState createState() => _LoginPageState();
}

class _LoginPageState extends State<LoginPage> {
  Duration get loginTime => Duration(milliseconds: 2250);

  Future<String> _authUserSignUp(LoginData data) {
    return Future.delayed(loginTime).then((_) async {
      try {
        await Provider.of<Auth>(context, listen: false)
            .signup(data.name, data.password);
      } catch (err) {
        print(err);
        return err.toString();
      }
      return null;
    });
  }

  Future<String> _authUserLogin(LoginData data) {
    return Future.delayed(loginTime).then((_) async {
      try {
        await Provider.of<Auth>(context, listen: false)
            .login(data.name, data.password);
      } catch (err) {
        print(err);
        return err.toString();
      }
      return null;
    });
  }

  Future<String> _recoverPassword(String name) {
    print('Name: $name');
    return Future.delayed(loginTime).then((_) {
```

```
      if (!users.containsKey(name)) {
        return 'Username not exists';
      }
      return null;
    });
  }

  @override
  Widget build(BuildContext context) {
    return FlutterLogin(
      title: 'ECORP',
      // logo: 'assets/images/ecorp-lightblue.png',
      onLogin: _authUserLogin,
      onSignup: _authUserSignUp,
      onSubmitAnimationCompleted: () {
        Provider.of<Auth>(context, listen: false).tempData();
      },
      onRecoverPassword: _recoverPassword,
    );
  }
}
```